EngOpt 2008 - International Conference on Engineering Optimization
Rio de Janeiro, Brazil, 01 - 5 June 2008.

Project Management – Multiple Resources Allocation

Anabela P. Tereso* **Madalena M. Araújo†** **Rui S. Moutinho‡** **Salah E. Elmaghraby§**

*anabelat@dps.uminho.pt †mmaraujo@dps.uminho.pt ‡rumout@gmail.com §elmaghra@eos.ncsu.edu
*†‡: Departamento de Produção e Sistemas
Universidade do Minho
4800-058 Guimarães
PORTUGAL

§: North Carolina State University
Raleigh, NC 27695-7906
USA

1. Abstract

Given a project network under stochastic conditions, the goal is to determine the optimal resource allocation to the activities in order to minimize the total project cost. This cost includes the resource cost and the tardiness cost. In this work we consider the multiple resources case, which is an extension of the models previously developed by the first author and other researchers, considering a single resource. We assume that all the resources are independent and abundant.

The work consists mainly of two parts: formalization of the new models, and their implementation in Java. In order to formalize the models, it was necessary to establish an allocation strategy for the multiple resources. This is required to ensure the desired equality of expected durations yielded by each resource in the same activity. We study four different allocation strategies: two of them are derived from the stochastic nature of the work content by equalizing the expected durations, thus determining the allocation vectors; and the other two go down to the level of all possible values to devise an allocation method (among all the allocation vectors, selects those leading to equal expected durations). Then the probability distributions of the variables required for analysis and evaluation were determined.

Although the research has covered four strategies, one proved to be inferior compared to the others, and another was too complex to be easily implemented. The remaining two are strong rivals with neither dominating the other, and one of them was arbitrarily chosen for implementation.

The implementation covers three algorithms: Dynamic Programming Algorithm, Electromagnetic Algorithm and Evolutionary Algorithm. Concurrent programming was exploited to enhance performance. We report on the performance of our application over a representative set of project networks.

2. Keywords: Project Management and Scheduling, Stochastic Activity Networks, Resource Allocation, Multiple Resources, Dynamic Programming, Electromagnetic Algorithm, Evolutionary Algorithm

3. List of Acronyms

PLC: *Project Live Cycle*
AoA: *Activity-on-Arc*
DP: *Dynamic Programming*
GOA: *Global Optimization Algorithm*
EMA: *Electromagnetic Algorithm*
EVA: *Evolutionary Algorithm*
RCPSP: *Resource Constraint Project Scheduling Problem*
DPA: *DP-based Algorithm*
DAG: *Directed Acyclic Graph*
SRPCO: *Single Resource Project Cost Optimization*
MRPCO: *Multiple Resources Project Cost Optimization*
QORAS: *Quantity Oriented Resource Allocation Strategy*
DORAS: *Duration Oriented Resource Allocation Strategy*
WBRAS: *Waste Balance Resource Allocation Strategy*
DC-Pair: *Duration-Cost Pair*
OF: *Objective Function*

4. Introduction and Review of Previous Work

This paper addresses the work first described on [1], and gives an overview of the developments to date on the expansion of the previous work to the multiple resources scenario.

We are concerned with the *planning* stage of the PLC (*Project Live Cycle*). Given a project which activities are represented in an AoA (*Activity-on-Arc*) network; we wish to evaluate the optimal allocation vector so as to minimize the total project cost. This latter is composed of two parts: the cost of the resource usage and the penalty for defaulting on a prescribed due date. Furthermore, the activities are assumed to be multimodal and stochastic in nature.

The minimization of project's total cost under stochastic conditions has been addressed before. These contributions varied from the initial DP (*Dynamic Programming*) oriented approach (on MATLAB) to the application of global optimization algorithms (on JAVA).

After the first DP implementation in MATLAB [2], a GOA (*Global Optimization Algorithm*), the EMA (*Electromagnetic Algorithm*) was implemented, still in MATLAB [3]. Seeking better performance, the first two approaches were re-implemented in JAVA [4, 5]. Then, another GOA, the EVA (*Evolutionary Algorithm*) was also applied to this problem [6]. The JAVA implementations exploited a distributed platform.

All these studies considered the projects with only one resource. Our work is still under the RCPSP (*Resource Constraint Project Scheduling Problem*) for minimization of total project cost but with an arbitrary number of resources. Thus, our work objectives are:

- To develop new models involving multiple resources projects. One for each optimization method: DPA (*DP-based Algorithm*), EMA and EVA;
- Implementation of the new models in JAVA; exploiting concurrent programming.

Furthermore, we shall assume the following premisses:

- The AOA representations will not have any dummy activities and are DAG (*Directed Acyclic Graph*) with only one initial node and one and only one final node;
- The project has a well determined set of resources available;
- The project activities can consume only a subset of the project resources;
- The quantity cost per unit is fixed. That means that the cost of the resources consumption do not changes during the execution of the project, e.g., interests according to delays.
- The resources are independent from each other, meaning that any resource may be used concurrently with others without any restrictions;
- The resources are abundant. There are no allocation restrictions concerning either concurrent or sequential activities.

5. Allocation Strategies

On SRPCO (*Single Resource Project Cost Optimization*) models so far implemented, the behavior of an activity was fully described by its single resource. However, on the new MRPCO (*Multiple Resources Project Cost Optimization*) models we need to describe that behavior with each resource. So, each resource will have its own work content and allocation constraints according to each activity's needs.

5.1. The Impact of Resource Multiplicity

The extension of the project cost evaluation from the SRPCO model is quite straightforward. The multiplicity of resources simply induces a sum of the allocation cost of each resource of each activity. Thus, the total project cost C is

$$\mathrm{C} = \mathcal{E}\left[\sum_{a\in \mathrm{A}}\sum_{r\in \mathrm{R}_a}\left(c_r \times \mathrm{x}_r^a \times W_r^a\right) + c_L \times \max\left(0, \Upsilon_n - T\right)\right] \tag{1}$$

where the following notation applies:

A: set of project activities;
R_a: project resources subset needed by activity a
c_r: quantity cost per unit of resource r
x_r^a: allocated quantity of resource r on activity a
$W_r^a \sim \mathrm{Exp}\left(\lambda_r^a\right)$: Work content of resource r on activity a
Υ_n: Evaluated realization time of last node
T: Schedule project realization time

To each resource allocation to an activity is associated an individual duration Y_r^a evaluated similar to the SRPCO model.

$$Y_r^a = \frac{W_r^a}{\mathrm{x}_r^a} \tag{2}$$

The actual activity duration – Y_a – is, therefore, the maximum of those individual ones.

$$Y_a = \max_{r\in \mathrm{R}_a}\left(Y_r^a\right) \tag{3}$$

Clearly it makes little sense to expend more of a resource (and incur a higher cost) to have the activity duration under this resource less than its duration under any other resource. Thus, it is desired to have

$$Y_i^a = Y_j^a,\ \forall i, j \in \mathrm{R}_a \tag{4}$$

or, since there are random variables involved,

$$\mathcal{E}\left[Y_i^a\right] = \mathcal{E}\left[Y_j^a\right], \forall i, j \in \mathrm{R}_a \tag{5}$$

To ensure allocation vectors leading to the desired equality, at least three strategies can be devised.

5.2. Quantity Oriented Strategy

The QOras (*Quantity Oriented Resource Allocation Strategy*) starts from the equality of individual durations in expectation and rearranges the equation so that the resources are all expressed relative to one of them, which shall be referred to as the "base" resource. This is possible because when considering expectations there are no longer random variables to deal with and one can easily transform the allocation problem of several resources into just one. The remaining allocations are immediately known through knowledge of the "base" resource allocation.

Despite its simplicity and intuitive appeal, this strategy needs frequent corrections to the allocations in order to ensure that they remain in their feasible regions and the proportionality relations between them are preserved. This corrective mechanism becomes increasingly complex and hard to implement as the number of resources per activity increases. Furthermore, this strategy fails in situations where the desired equality is impossible to realize.

5.3. Duration Oriented Strategy

This approach – DOras (*Duration Oriented Resource Allocation Strategy*) – depends on sampling the work content. Then, the samples of the individual durations are determined by evaluating the possible durations according to the feasible allocation values.

The approach proceeds by evaluating the possible common durations via intersecting all the combinations of the sampled individual durations. Then, it filters those intervals leaving only the larger ones (those that yielded more feasible resources). If the filtering results in just one interval, we have in hand the possible equal durations yielded by all resources, and the desired equality is satisfied. Else, it chooses the interval with the higher value. Either way, the durations of the selected interval are used to retrieve the allocation vector (of the involved resources) leading to each of them. Those resources not contributing to the selected interval, are put equal to their minimum values.

This strategy is too complex to be implemented. Its algorithms experience exponential growth in both number of activities and resources. However, it copes rather well with any number of resources and with the cases when the equality of individual durations is impossible to achieve.

5.4. Waste Balanced Strategy

In the WBras (*Waste Balance Resource Allocation Strategy*) we establish a mechanism that ensures always equal individual durations; see Figure 1.

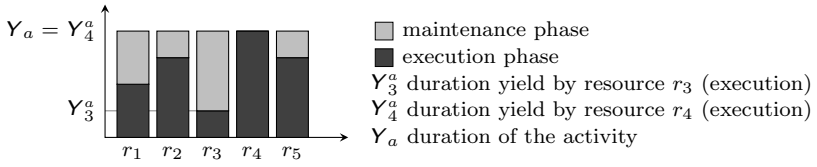

Figure 1: WBras: Activity duration with balanced individual durations

If we designate the time from the moment when a resource ceases to be needed to the end of the activity execution as "maintenance time", then we are able to quantify the inequality of individual durations. Each resource that lies unused may carry maintenance fees: storage, lifetime decline; etc. Thus, we may conceptualize an activity as being a set of sub-activities in parallel, one per each resource, and composed of an execution phase followed by a maintenance phase. The maintenance duration for a resource $r \in \mathrm{R}_a$ is simply the difference of the activity duration and the individual duration yielded by r alone. In Figure 2 on the next page we can see this conceptualization for a single activity.

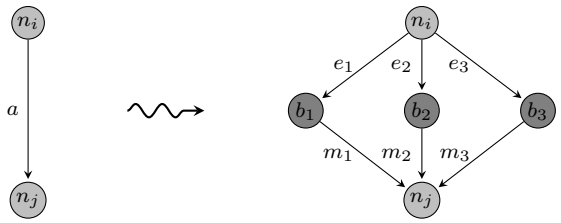

Figure 2: WBRAS: Extended activity concept (3 sub-activities where e_i are the execution phases; m_i the maintenance phases and b_i the balance nodes)

By having that maintenance duration, we can apply to it a (time) unit cost; and so we can incorporate the maintenance cost into the resource allocation cost (the previously defined one). This way we ensure optimal allocations through the project total cost. It is important to notice that in order for this strategy to be correctly applied, the project must be explicitly "maintenance aware". By this we mean that the actual project cost to be optimized must be the sum of the project cost of allocations *including the maintenance costs*. If we then discard the maintenance portion we would not know whether or not the allocation cost is the optimal. In fact, this strategy induces a new factor that characterizes the project.

6. Optimization Models for Multiple Resources

We base our models on the WBRAS. From this strategy we get the maintenance durations depending on the activity duration, which is a random variable. Thus, the maintenance duration is a random variable and, consequently, the maintenance cost is also a random variable.

Since the DP-based approach is based upon decisions made accordingly to stages, we must take care of both the randomness nature and the dependency of the maintenance duration/cost. On the other hand, the GOA-based model is quite straightforward. In the next two sections we explain the two models.

6.1. DP-Based Model

This section assumes previous background knowledge of the references cited in section 4 on page 1.

All the random variables present on the model are, ultimately, dependent on the work content of the resources. So, by creating representative samples for those work contents, we can derive samples for all the other random variables.

A representative sample of size n of a work content $W \sim \text{Exp}(\lambda)$ is created by evaluating the sample values ω_i, $i = 1, \ldots, n$ where each one has probability equal to $1/n$ and

$$\mathcal{E}\left[\{\omega_i \,|\, i = 1, \ldots, n\}\right] = \frac{\omega_1 + \cdots + \omega_n}{n} = \frac{1}{\lambda} = \mathcal{E}\left[W\right] \tag{6}$$

One way of describing the relationship between an activity duration and its maintenance cost is through the following set, for an arbitrarily fixed activity allocation vector:

$$\Phi_a = \left\{ \left(\psi, \sum_{i \in \mathrm{R}_a} \left(s_i \left(\psi - \psi_i \right) \right) \right) \middle| \psi = \max_{i \in \mathrm{R}_a} \left(\psi_i \right), \forall \psi_i \in \mathcal{Y}_i^a \right\} \tag{7}$$

Each $\mathcal{Y}_i^a$ is a sample of an individual duration resulting from the allocation of each resource i to activity a. ψ represents a sample value of the maximum of the individual durations, hence a sample value of Y_a. The s_i values are the maintenance cost per unit time of each resource. Thus, each element of Φ_a associates an activity duration sample value (the first component of the element) with the maintenance cost (the second component of the same element). Each one of elements in Φ_a will be named as DC-Pair (*Duration-Cost Pair*) and the ensemble represents the sample values of the relation between the activity duration and the maintenance cost. Each pair has the probability of realization equal to the probability of the first component because of the strict relationship between the two components of each pair in Eq. (7).

The sample Φ_a yields a distribution of the two parameters involved in the DC-Pair. This distribution approximates the real distribution of these parameters and the approximation improves with increase in the sample size.

For the evaluation of the realization times of the nodes on the AoA representation, we need to cope with the newly defined Φ_a, $\forall a \in \mathrm{A}$. More precisely, the realization of a node must contemplate the new factor – maintenance cost. Consequently, the states now must be vectors of DC-Pair instead of only realization times. Thus, given such a state[1] s_l we want to estimate the realization of a node. To achieve that estimation we must know the distributions for the activities which connect the state nodes to the target one. But then we need to specify that transition. Thus when, say, we go from state node n to node k through activity a we must add the activity duration to the realization time of the node n and store the maintenance cost of the activity, hence the following equation:

$$\Phi_{[n,a]} = \{(\alpha_n + \alpha_a, \beta_a) | \forall\, (\alpha_a, \beta_a) \in \Phi_a\}\,,\ (\alpha_n, \beta_n)\ \text{ state on node } n \tag{8}$$

Notice that we do not want to add the maintenance cost of the state node. This will ensure that no cost is duplicated further on.

The actual estimation on the realization of a node will be determined from one of two scenarios. If the node is already contemplated on the state, then nothing is to be determined (its realization is already known). Else, we must set its realization time as the maximum of all the "arriving" activities contributions (durations) and add all the maintenance costs from them. Already on Eq. (8) we described the transition from a state node through an (arriving) activity to another node. Now, we must have a function that wraps those transitions in order to describe the above behavior:

$$\mathtt{DCmax}\left(\Phi_1, \ldots, \Phi_n\right) = \left\{ \left(\max_{i=1}^{n} \left(\alpha_i\right)\,, \sum_{i=1}^{n} \beta_i \right) \middle| \forall\, (\alpha_i, \beta_i) \in \Phi_i \right\} \tag{9}$$

Hence, the estimate realization of a node k (denoted as Υ_k), given a state s_l is:

$$\Upsilon_k = \begin{cases} \{(\alpha_k, \beta_k)\} & ,\ \text{node } k \text{ contemplated on state } s_l \\ \underset{(n,a)\in P}{\mathtt{DCmax}} \left(\Phi_{[n,a]}\right) & ,\ \text{otherwise} \end{cases} \tag{10}$$

where (α_k, β_k) is the state on node k case it is contemplated on state s_l and P is the AoA network part preceding node k, with each element representing a predecessor node and the activity on the arc connecting it to node k.

The cost of resource utilization in an activity (referred to as "the quantity cost of an activity") is evaluated differently from the maintenance cost. So, let C^a_{Q} be the quantity cost of activity a:

$$C^a_{\mathrm{Q}} = \sum_{r \in \mathrm{R}_a} \left(c_r \times \mathrm{x}^a_r \times W^a_r\right) \tag{11}$$

Let $\mathcal{F}$ denote the set of all "fixed activities" (those which resources' allocation are not decision variables but rather fixed), the resource quantity cost of the fixed activities (denoted by $\mathrm{rcf}_{\mathrm{Q}}$) is given by

$$\mathrm{rcf}_{\mathrm{Q}} = \mathcal{E}\left[\sum_{a \in \mathcal{F}} C^a_{\mathrm{Q}}\right] = \sum_{a \in \mathcal{F}} \sum_{r \in \mathrm{R}_a} \left(c_r \times \mathrm{x}^a_r \times \mathcal{E}\left[W^a_r\right]\right) \tag{12}$$

Finally, indexing the decision activities by their stage, the first stage formula for the DP-based model is:

$$f_1(s_1|\mathcal{F}) = \mathrm{rcf}_{\mathrm{Q}} + \min_{\mathrm{X}_1} \left(\mathcal{E}\left[C^1_{\mathrm{Q}} + \mathcal{E}\left[\mathrm{sumpair}\left(c_L \dot{\times}\, U\right)\right]\right]\right) \tag{13}$$

where U reflects the tardiness of the project:

$$U = \mathtt{DCmax}\left(\overline{0},\, \Upsilon_n - \overline{T}\right) \tag{14}$$

where Υ_n represents the realization of the last node and $\Upsilon_n - \overline{T}$ means subtracting the constant T to all of the first components of the elements of Υ_n. The notation $\overline{h}$ when h is a constant refers to the

[1] We use the same notation for both maintenance cost per unit time and states. That should not pose a problem as the correct meaning shall be transparent from the context.

DC-Pair distribution composed by only one element (of probability 1) in which the first component is equal to h and the second equal to zero. Thus

$$\overline{0} = \{(0,0)\} \qquad \overline{T} = \{(T,0)\} \tag{15}$$

The function $\dot{\times}$ multiplies c_L with the first component of each element of U; resulting in a new distribution whose elements associate the tardiness cost (fist component) to the maintenance cost (second component). Then, the function sumpair adds those two costs on each element; thus resulting in a distribution whose elements are the sums. Both $\dot{\times}$ and sumpair do not alter the probability of the initial pairs. So, the probability of each sum is equal to the one of the original pair.

The formula for stages $k > 1$ is[2]:

$$f_k(s_k|\mathcal{F}) = \sum_{(\alpha,\beta)\in s'_k} \beta + \min_{\mathrm{X}_k} \left(\mathcal{E}\left[C^k_{\mathrm{Q}} + \mathcal{E}\left[f_{k-1}(s_{k-1}|\mathcal{F}) \right] \right] \right) \tag{16}$$

where $s'_k = s_k \setminus s_{k-1}$. Then, the main formula of the model is:

$$f(s_K = 0) = \min_{\mathcal{F}} \left(f_K(s_K|\mathcal{F}) \right) \tag{17}$$

where $s_K = 0$ means the initial state. This is associated with the initial node and represents zero duration and zero maintenance cost.

6.2. GOA-Based Model

The GOA implementations will rely on the *Monte Carlo Simulation* over the work contents. Thus, for the evaluation of the total project cost it is only required to process the realization (execution and maintenance durations) of each node; evaluate the total maintenance cost and add it to the tardiness cost and quantity allocation cost. Because of the random nature of the choice of values for the work contents, we no longer have to deal with random variables. Thus the model is linear.

Let us denote by C^a_{M} the maintenance cost associated of the activity $a \in \mathrm{A}$ evaluated as:

$$C^a_{\mathrm{M}} = \sum_{r\in \mathrm{R}_a} (s_r \times (Y_a - Y^a_r)) \tag{18}$$

Recall from above evaluation of C^a_{Q} and Υ_n that both involve deterministic values in this context. The GOA will minimize the following function:

$$f = \sum_{a\in \mathrm{A}} \left(C^a_{\mathrm{Q}} + C^a_{\mathrm{M}} \right) + c_L \times \left(\max\left(0, \Upsilon_n - T\right) \right) \tag{19}$$

7. Implementation on Java

The models were implemented in JAVA 1.6. A single command line application was created to deal with all the models and runtime specifications: optimization engine; with or without parallel exploitation; engine configurations; etc.

7.1. AoA Representation

A complete library for AoA network manipulation, including basic construction, was implemented to deal with the new activities with multiple resources.

7.2. Distributions

To help work with distributions, a complete library of classes was implemented. These can sample a continuous distribution or even create a partition of a given interval. For example, one of the classes can create sample distributions of the exponential distribution with arbitrary number of sample values. Also, the classes allow operations like the sum or maximum of distributions.

[2]For ease of notation we make an abuse of language: despite s_k being a vector we address each of its components as it were a set; hence the use of $\in$ and set difference.

7.3. Combinatorics
The frequent need for walking through all the possible combinations of a number of (partitioned) variables led to the creation of classes providing linear addressing of those. Thus, instead of computing complex nesting "for" cycles, we can simply walk from the first combination to the last, deterministically and linearly. Also it is possible to process asynchronously subsets of the combinations: useful for distributed systems.

7.4. DP Model Specifics
The implementation of the DP model follows the natural recurrence of the model. This brings weaker performance to the application but achieves more accurate results.

7.5. GOA Model Specifics
Both EMA and EVA use entities mapped as vectors. This causes a problem when applied to activities with multiple resources. Thus, instead of representing each activity on each vector component, we concatenate all the allocation vectors of the activities and mark the start and end locations. The GOA proceeds like usual and at the end we re-assemble the activities allocation vectors.

The EMA algorithm was adapted from the original imperative-oriented form to an object-oriented form; allowing easy addition of improvements like the concurrent programming. The EVA algorithm was also tuned up.

7.6. Parallelism Exploitation
The DPA can be divided in several independent tasks: one per each allocation combination of the fixed activities. Also, the GOA involve several replication runs before giving the final (better) result. Thus, we enable these tasks to be processed concurrently.

For the GOA, usually the evaluation of the OF (*Objective Function*) value is not immediately required as other operations need to be performed first. For example, in the EMA the computation of the forces do not take all the OF values at once; and on the EVA the OF values are only required at the time of selection. We exploited this fact by enabling the OF value evaluation to be asynchronous. That is, instead of evaluating one-at-a-time without doing nothing else; it is possible to instruct each entity to evaluate its own OF value while the main process continues the underlying operations. Later, when the OF values are actually needed they will (hopefully) be ready; otherwise the main process waits (but only as needed, not always).

8. Results
The implementation was tested on Windows Vista with Java virtual machine 1.6 at 64 bits. The processor was a Duo T7300 and 2GB of RAM. All the tests still running after 5 hours were canceled.

All the tested projects have all the resources with the allocation interval set as $[0.5, 1.5]$. The tests cover the application of the DPA, EMA and EVA at both single-thread and multi-thread mode. Also, two different configurations were applied on the EMA and EVA: $k = 500$ and $k = 5000$. K is the number of random values used for the work contents (Monte Carlo Simulation). The DPA was performed with work content sample sizes of 4 elements.

The results between single-threaded and multi-threaded runs were consistent with each other. For brevity, we only present one allocation vector (the one resulting from the single-threaded run) for both cases. The allocation vectors are presented respecting the natural order of the resources indexation. For example, if an activity a has the resources r_1, r_3 and allocation vector $\mathrm{X}_a = (1, 2)$ then $\mathrm{x}_1^a = 1$, $\mathrm{x}_3^a = 2$. The DPA outputs allocation vectors for the first decision activity and for all the fixed activities. The difference between the decision and fixed allocation vectors is made through the use of $\mathrm{X}_a = \{1, 2\}$ for the decision activities and $\mathrm{X}_a = (1, 2)$ to all the others.

In the topology figures, the thicker edges signal a decision activity.

8.1. Test A

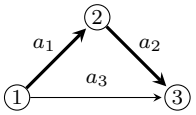

Figure 3: `AoA` Topology with 3 activities and 3 nodes

Table 1: Project A.a – Single Resource

$T = 16$		a_1	a_2	a_3		c_r	s_r
$c_L = 2$	λ_1^a	0.2	0.1	0.07	r_1	1.0	0.0

Table 2: Project A.a – Runtime Statistics

	`DPA`	`EMA` ($k = 500$)	`EMA` ($k = 5000$)	`EVA` ($k = 500$)	`EVA` ($k = 5000$)
Linear	0:00:00.203	0:00:01.450	0:00:05.445	0:00:01.294	0:00:04.977
Concurrent	0:00:00.171	0:00:00.889	0:00:03.385	0:00:01.450	0:00:03.728

Table 3: Project A.a – Allocation Results

	a_1	a_2	a_3	`OF` value
`DPA`	$\{1.0\}$		(1.0)	43.7
`EMA`($k = 500$)	(0.911)	(0.888)	(0.851)	37.531
`EMA`($k = 5000$)	(0.872)	(0.911)	(0.854)	38.988
`EVA`($k = 500$)	(0.874)	(0.887)	(0.84)	37.291
`EVA`($k = 5000$)	(0.892)	(0.879)	(0.841)	38.68

Table 4: Project A.b – Multiple Resources (configuration one)

$T = 41$		a_1	a_2	a_3		c_r	s_r
$c_L = 2$	λ_1^a	0.07		0.09	r_1	1.0	1.0
	λ_2^a	0.1	0.04		r_2	1.1	2.0
	λ_3^a	0.2			r_3	1.0	0.5

Table 5: Project A.b – Runtime Statistics

	`DPA`	`EMA` ($k = 500$)	`EMA` ($k = 5000$)	`EVA` ($k = 500$)	`EVA` ($k = 5000$)
Linear	0:00:00.842	0:00:02.910	0:00:12.745	0:00:02.106	0:00:09.790
Concurrent	0:00:00.827	0:00:01.436	0:00:07.285	0:00:01.779	0:00:06.193

Table 6: Project A.b – Allocation Results

	a_1	a_2	a_3	`OF` value
`DPA`	$\{1.5, 1.0, 1.0\}$		(0.5)	113.147
`EMA`($k = 500$)	$(1.164, 0.814, 0.5)$	(1.096)	(0.5)	84.742
`EMA`($k = 5000$)	$(1.223, 0.856, 0.501)$	(1.102)	(0.501)	88.108
`EVA`($k = 500$)	$(1.195, 0.836, 0.5)$	(1.087)	(0.5)	84.335
`EVA`($k = 5000$)	$(1.118, 0.782, 0.5)$	(1.113)	(0.5)	87.234

Table 7: Project A.c – Multiple Resources (configuration two)

$T = 61$		a_1	a_2	a_3		c_r	s_r
$c_L = 2$	λ_1^a	0.1	0.03		r_1	1.0	1.0
	λ_2^a	0.2	0.06	0.07	r_2	1.1	2.0
	λ_3^a		0.03	0.09	r_3	1.0	0.5
	λ_4^a	0.04	0.04	0.07	r_4	2.0	0.1

Table 8: Project A.c – Runtime Statistics

	`DPA`	`EMA` ($k = 500$)	`EMA` ($k = 5000$)	`EVA` ($k = 500$)	`EVA` ($k = 5000$)
Linear	> 5h	0:00:07.660	0:00:58.812	0:00:03.978	0:00:29.160
Concurrent	> 5h	0:00:04.410	0:00:31.153	0:00:03.339	0:00:16.680

Table 9: Project A.c – Allocation Results

	a_1	a_2	a_3	OF value
DPA				*aborted*
EMA($k = 500$)	$(0.568, 0.5, 1.421)$	$(1.072, 0.547, 1.072, 0.804)$	$(0.64, 0.504, 0.64)$	274.8
EMA($k = 5000$)	$(0.557, 0.5, 1.393)$	$(1.053, 0.536, 1.054, 0.79)$	$(0.621, 0.507, 0.622)$	281.107
EVA($k = 500$)	$(0.533, 0.5, 1.331)$	$(1.048, 0.542, 1.048, 0.786)$	$(0.523, 0.5, 0.523)$	269.528
EVA($k = 5000$)	$(0.507, 0.5, 1.267)$	$(1.058, 0.533, 1.058, 0.805)$	$(0.542, 0.5, 0.567)$	276.860

8.2. Test B

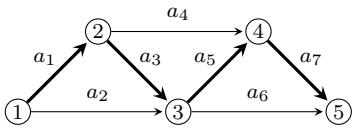

Figure 4: AoA Topology with 7 activities and 5 nodes

Table 10: Project B.a – Single Resource

$T = 66$, $c_L = 5$

	a_1	a_2	a_3	a_4	a_5	a_6	a_7
λ_1^a	0.08	0.06	0.09	0.05	0.07	0.03	0.04

	c_r	s_r
r_1	1.0	0.0

Table 11: Project B.a – Runtime Statistics

	DPA	EMA ($k = 500$)	EMA ($k = 5000$)	EVA ($k = 500$)	EVA ($k = 5000$)
Linear	0:00:50.607	0:00:04.461	0:00:34.351	0:00:03.270	0:00:18.189
Concurrent	0:00:30.467	0:00:02.808	0:00:16.707	0:00:02.714	0:00:11.123

Table 12: Project B.a – Allocation Results

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	OF value
DPA	$\{1.5\}$	(1.0)		(1.0)		(1.5)		205.541
EMA($k = 500$)	(1.364)	(0.87)	(1.117)	(0.882)	(1.127)	(1.053)	(1.319)	209.633
EMA($k = 5000$)	(1.265)	(0.872)	(1.209)	(0.894)	(1.093)	(1.027)	(1.301)	218.927
EVA($k = 500$)	(1.415)	(0.939)	(1.249)	(0.891)	(1.064)	(1.024)	(1.316)	209.182
EVA($k = 5000$)	(1.432)	(0.925)	(1.201)	(0.886)	(1.075)	(1.055)	(1.374)	216.076

Table 13: Project B.b – Multiple Resources (configuration one)

$T = 129$, $c_L = 5$

	a_1	a_2	a_3	a_4	a_5	a_6	a_7
λ_1^a	0.02	0.04			0.03	0.04	0.07
λ_2^a	0.08		0.04	0.06	0.05		0.08

	c_r	s_r
r_1	1.0	1.0
r_2	1.1	2.0

Table 14: Project B.b – Runtime Statistics

	DPA	EMA ($k = 500$)	EMA ($k = 5000$)	EVA ($k = 500$)	EVA ($k = 5000$)
Linear	0:42:07.293	0:00:07.846	0:01:05.208	0:00:04.524	0:00:35.318
Concurrent	0:25:52.130	0:00:04.337	0:00:33.150	0:00:03.572	0:00:19.516

Table 15: Project B.b – Allocation Results

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	OF value
DPA	$\{1.5, 0.75\}$	(1.0)		(0.5)		(1.0)		393.297
EMA($k = 500$)	$(1.5, 0.5)$	(0.5)	(1.5)	(0.5)	$(1.339, 0.804)$	(0.658)	$(1.104, 0.966)$	361.883
EMA($k = 5000$)	$(1.5, 0.5)$	(0.5)	(1.5)	(0.5)	$(1.272, 0.763)$	(0.643)	$(1.146, 1.002)$	376.068
EVA($k = 500$)	$(1.5, 0.5)$	(0.5)	(1.499)	(0.5)	$(1.274, 0.764)$	(0.659)	$(1.22, 1.068)$	360.236
EVA($k = 5000$)	$(1.5, 0.5)$	(0.5)	(1.484)	(0.5)	$(1.282, 0.769)$	(0.651)	$(1.191, 1.043)$	374.425

Table 16: Project B.c – Multiple Resources (configuration two)

$T = 155$, $c_L = 5$

	a_1	a_2	a_3	a_4	a_5	a_6	a_7
λ_1^a		0.04			0.02	0.04	0.07
λ_2^a	0.02			0.07		0.04	
λ_3^a			0.03	0.09	0.05	0.024	

	c_r	s_r
r_1	1.0	1.0
r_2	1.1	2.0
r_3	1.0	0.5

Table 17: Project B.c – Runtime Statistics

	DPA	EMA ($k = 500$)	EMA ($k = 5000$)	EVA ($k = 500$)	EVA ($k = 5000$)
Linear	> 5h	0:00:10.904	0:01:32.493	0:00:05.522	0:00:48.126
Concurrent	> 5h	0:00:05.709	0:00:49.470	0:00:03.915	0:00:24.757

Table 18: Project B.c – Allocation Results

	a_1	a_2	a_3	a_4	a_5	a_6	a_7	OF value
DPA								*aborted*
EMA($k = 500$)	(1.5)	(0.5)	(1.5)	$(0.667, 0.519)$	$(1.436, 0.575)$	$(0.564, 0.564, 0.939)$	(1.429)	429.936
EMA($k = 5000$)	(1.5)	(0.5)	(1.499)	$(0.656, 0.51)$	$(1.453, 0.582)$	$(0.575, 0.575, 0.958)$	(1.45)	447.548
EVA($k = 500$)	(1.494)	(0.5)	(1.499)	$(0.679, 0.529)$	$(1.452, 0.681)$	$(0.575, 0.575, 0.958)$	(1.421)	433.6
EVA($k = 5000$)	(1.5)	(0.5)	(1.5)	$(0.653, 0.51)$	$(1.427, 0.571)$	$(0.548, 0.548, 0.914)$	(1.359)	441.731

9. Discussion and Conclusion

For small networks with few resources, the DPA is acceptable and is even faster than the GOA with large samples (high k). But, as soon as the networks grow by the number of resources the DPA quickly rises to run times far beyond 5 hours, while the GOA stay between few seconds to a couple of minutes; even with very large k. The EVA is the algorithm that results in the best performance.

The performance of all algorithms reflects the benefits of the concurrent programming.

The resulting allocations and OF values obtained by the several algorithms are consistent with each other.

The large $k = 5000$ showed no improvement on the results produced with the GOA.

We conclude that the simple (recurrent) implementation of the DPA is not suitable for practical tests, while the GOA represent a good alternative in both performance and OF values. A new implementation of the DPA is, therefore, desirable in order to expand the spectrum of tested projects.

10. References

[1] R. Moutinho. Gestão de Projectos – Alocação de Múltiplos Recursos. Relatório de estágio, Universidade do Minho, December 2007.

[2] A. P. Tereso, M. M. T. Araújo, and S. E. Elmaghraby. Adaptive Resource Allocation in Multimodal Activity Networks. *International Journal of Production Economics*, 92(1):1–10, November 2004.

[3] A. P. Tereso, M. M. T. Araújo, and S. E. Elmaghraby. The Optimal Resource Allocation in Stochastic Activity Networks via the Electromagnetism Approach. *Ninth International Workshop on Project Management and Scheduling (PMS'04)*, April 2004.

[4] A. P. Tereso, J. R. M. Mota, and R. J. T. Lameiro. Adaptive Resource Allocation to Stochastic Multimodal Projects: A Distributed Platform Implementation in Java. *Control and Cybernetics Journal*, 35(3):661–686, 2006.

[5] A. P. Tereso, R. A. Novais, and M. M. T. Araújo. The Optimal Resource Allocation in Stochastic Activity Networks via the Electromagnetism Approach: A Platform Implementation in Java. Reykjavíc, Iceland, July 2006. 21st European Conference on Operational Research (EURO XXI). Submitted to the "Control and Cybernetics Journal" (under revision).

[6] A. P. Tereso, L. A. Costa, R. A. Novais, and M. T. Araújo. The Optimal Resource Allocation in Stochastic Activity Networks via the Evolutionary Approach: A Platform Implementation in Java. Beijing, China, May 30 – June 2 2007. International Conference on Industrial Engineering and Systems Management (IESM' 2007). Full paper published in the proceedings (ISBN 978-7-89486-439-0) and submitted to the "Computers and Industrial Engineering".